

AD-A267 170



**DEPARTMENT OF DEFENCE**

**DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION**

**AERONAUTICAL RESEARCH LABORATORY**

**MELBOURNE, VICTORIA**

**DTIC**  
**ELECTE**  
**JUL 23 1993**  
**S A D**

Technical Note 17

**FLIGHT INSTRUMENT SOFTWARE FOR THE F/A-18  
RESEARCH SIMULATOR**

\*Original contains color  
plates: All DTIC reproduct-  
ions will be in black and  
white\*.

by

Y.Y. LINK

This document has been approved  
for public release and sale, its  
distribution is unlimited.

Approved for public release.

© COMMONWEALTH OF AUSTRALIA 1993

MARCH 1993

93 7 22 002

93-16628



**This work is copyright. Apart from any fair dealing for the purpose of study, research, criticism or review, as permitted under the Copyright Act, no part may be reproduced by any process without written permission. Copyright is the responsibility of the Director Publishing and Marketing, AGPS. Enquiries should be directed to the Manager, AGPS Press, Australian Government Publishing Service, GPO Box 84, CANBERRA ACT 2601.**

1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025

# DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

**DEPARTMENT OF DEFENCE  
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION  
AERONAUTICAL RESEARCH LABORATORY**

Technical Note 17

**FLIGHT INSTRUMENT SOFTWARE FOR THE F/A-18  
RESEARCH SIMULATOR**

by

Y.Y. LINK

**SUMMARY**

*The standby flight instruments are replicated on two high resolution raster displays for the F/A-18 research simulator. A software package was developed using NOVA\*CGI, an implementation of the CGI graphics standard, to control a MVME393 multi-channel graphics controller. The software is written in C for the Motorola MVME147 single board computer under the UNIX System VI68 operating system. Increased performance is achieved by decomposing the instrument dial pointers into a number of trapezoids, and storing these as device level command lists.*



© COMMONWEALTH OF AUSTRALIA 1993

**POSTAL ADDRESS:** Director, Aeronautical Research Laboratory,  
506 Lorimer Street, Fishermens Bend, 3207  
Victoria, Australia.

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and or Special
A-1	

## CONTENTS

### NOTATION

### LIST OF FIGURES AND TABLES

1.0 INTRODUCTION .....	1
2.0 SYSTEM .....	1
2.1 Hardware .....	1
2.2 Software.....	2
2.2.1 CGI .....	2
2.2.2 Device Level Command Lists .....	3
2.2.3 Drawing Method.....	3
2.2.4 Program Structure.....	4
2.2.5 Example File.....	4
2.2.6 Error Logging .....	5
3.0 INSTRUMENTS .....	5
3.1 Airspeed Indicator - airspeed.c.....	5
3.2 Altimeter - altimeter.c .....	5
3.3 Rate of Climb Indicator - climbrate.c.....	6
3.4 Attitude Indicator - horizon.c .....	6
3.5 Jettison Station Select - jetstat.c .....	7
3.6 Engine - engine.c .....	7
3.7 Fuel - fuel.c.....	8
3.8 Porting Code to IRIS .....	8
4.0 PERFORMANCE .....	9
4.1 Dial Pointer Drawing.....	9
4.2 Trapezoidal Decomposition of a Polygon .....	9
4.3 Overall Performance.....	10
5.0 CONCLUSION .....	10
ACKNOWLEDGEMENTS .....	10
REFERENCES .....	11
APPENDIX A - List of Modules and Functions .....	12
APPENDIX B - Trapezoidal Decomposition of a Dial Pointer .....	14
FIGURES .....	16

## NOTATION

CGI	-	Computer Graphics Interface
CL	-	Command List
ft/min	-	feet per minute
Hz	-	Hertz
lbs/h	-	pounds per hour
mba	-	millibars
psi	-	pounds per square inch
ms	-	milliseconds

## LIST OF FIGURES AND TABLES

Figure 1	-	Hardware Configuration
Figure 2	-	Connections and Virtual Devices
Figure 3	-	F/A-18 Main Instrument Panel
Figure 4	-	Flight Instrument Displays
Table 1	-	Flight Instrument Performance

## 1.0 INTRODUCTION

An F/A-18 flight simulator is being developed for research purposes for the Air Operations Division at ARL. Potential uses for this facility will be in human factors pilot workload studies, air combat studies, simulator effectiveness studies, and assessment of effectiveness and survivability of aircraft in ground attack missions.

The task requires visual replication of the F/A-18 cockpit. This document describes the computer software designed to replicate two of the instrument panels found in the F/A-18 cockpit.

The standby flight instruments, including the airspeed indicator, altimeter, rate of climb indicator and attitude indicator, are located on the right lower panel of the main instrument panel. The undercarriage, flap, engine and fuel instruments are located on the left lower panel of the main instrument panel (Figure 3).

The two panels are replicated on two high resolution raster displays (Figure 4). The graphics software is written in C using the NOVA\*CGI<sup>1</sup> software interface. The software is running on a Motorola MVME147 single board computer, under the UNIX System V/68 operating system, driving an MVME393 multi-channel graphics controller (Figure 1).

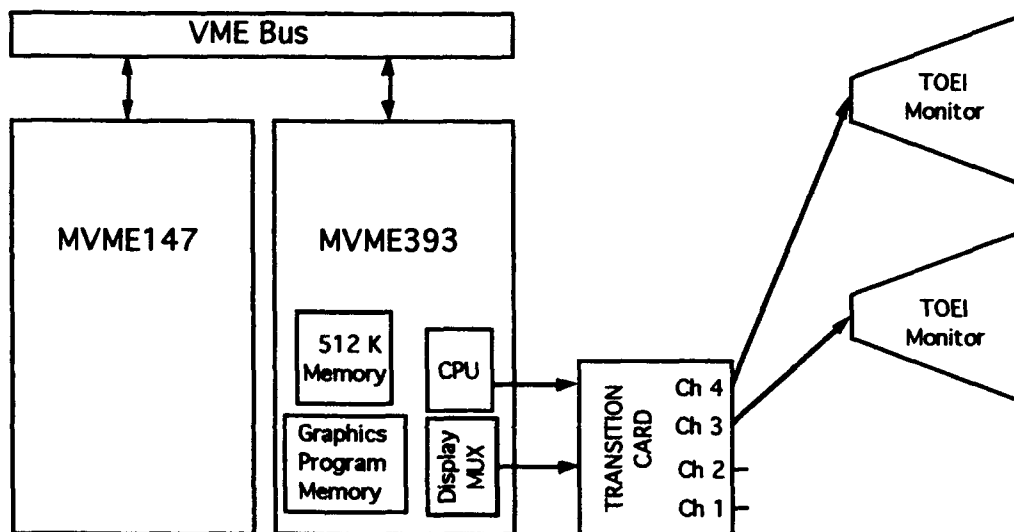


Figure 1: Hardware Configuration

## 2.0 SYSTEM

### 2.1 Hardware

The flight instrument software is run on a Motorola MVME147 VME module monoboard microcomputer, which includes an MC68030 microprocessor. The

<sup>1</sup> NOVA\*CGI is a registered trademark of Nova Graphics International Corporation.

MVME147 is operated as part of a VME bus system with another VME module, the MVME393 graphics controller.

The MVME393 graphics controller is a medium resolution, multi-channel graphics display controller. It contains dual independent processors, the MC68010 local processor and the TMS34010 graphics system processor. The graphics controller can drive up to eight bit-mapped displays up to a resolution of 1024 x 256 x 4-bit colour, four displays up to a resolution of 1024 x 512 x 4-bit colour, or two displays up to a resolution of 1024 x 1024 x 4-bit colour. The current configuration is four channels output, with a programmed resolution of 640 x 480 and 60 Hz update rate.

The two displays to be used in the cockpit are TOEI CDM-103 monitors. This is a 250 millimetre colour display monitor capable of displaying RGB input signals, at 640 x 480 resolution, and 60 Hz update rate.

One of the flight instrument displays has been ported to run on the Silicon Graphics IRIS 320 GTX. The code was converted to use the IRIS graphics library routines. (Section 3.9)

## 2.2 Software

### 2.2.1 CGI

Computer Graphics Interface (CGI) is a graphics standard developed by the International Standards Organisation. The implementation of the standard used to develop the flight instrument software is NOVA\*CGI V3.2. NOVA\*CGI provides a library of C language functions to perform the generation of graphics. For a detailed explanation of NOVA\*CGI and CGI the reader is referred to References [1] and [2].

NOVA\*CGI allows communication to be established with a drawing surface over a 'connection' (Figure 2). Each physical drawing surface requires at least one connection in order to establish communication. However, it is feasible to open more than one connection to any physical drawing surface.

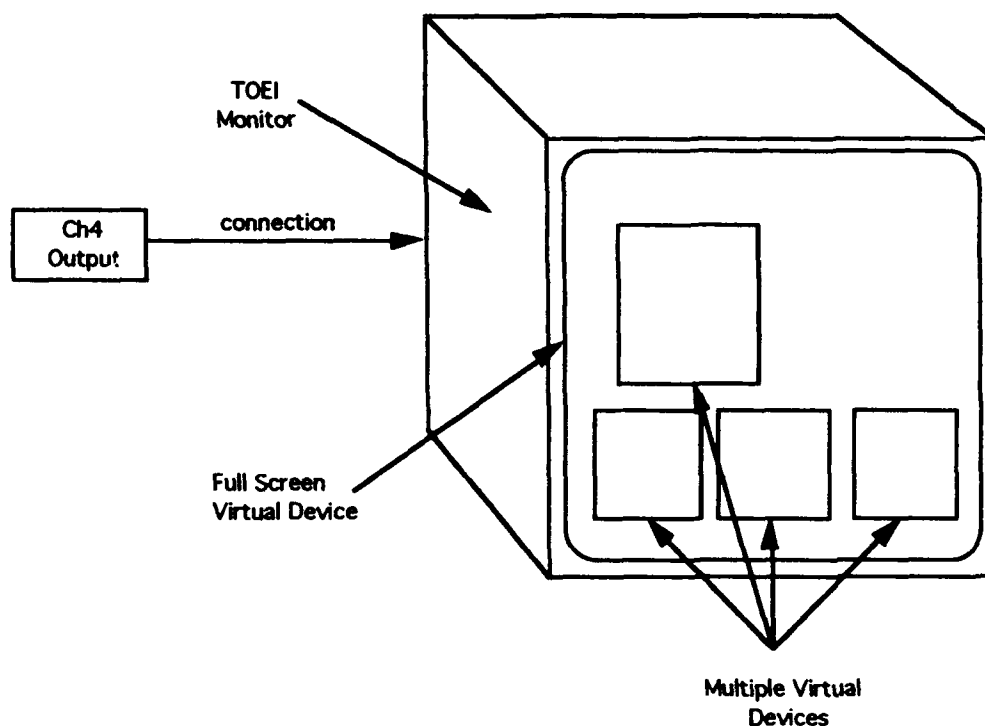
NOVA\*CGI allows the creation of multiple windows over one connection on a single physical drawing surface. These windows are defined by CGI as 'virtual devices'. It is obligatory to open a full screen virtual device in order to communicate with a drawing surface. Each virtual device has associated with it, its own state lists and coordinate system. The only resource that is shared between virtual devices is the colour table.

The ability to create virtual devices is convenient for modelling each of the instruments as an entirely separate entity. On the selection of a virtual device the graphic commands buffer for the previous virtual device is flushed. It can therefore be assumed that drawing commands will be executed in a sequential order<sup>2</sup>.

---

<sup>2</sup> In contrast to individual connections which do not flush the buffer (Ref. [1] p.28)





**Figure 2: Connections and Virtual Devices**

### 2.2.2 Device Level Command Lists

Device level command lists are a non-standard NOVA\*CGI extension to the CGI standard, and are intended to improve the performance of graphics applications. The CGI standard imposes additional overhead, which results in loss of performance in relation to the ability of the graphics hardware.

The Command List is a list of low level graphics commands. The graphics commands available are a subset of the standard graphics command set. Once a list of commands is built, it can be saved with a unique command list identifier, in the Interpreter's local memory. The Interpreter runs on the Graphics Program Memory of the MVME393 graphics card (Ref. [3]). This eliminates the need to download the commands on each invocation, and subsequently improves performance. There is a limit of 1024 command list identifiers, which limits the number of command lists to 1024.

A command list buffer consists of a sequence of opcodes and parameters (Ref. [1]), which are 16 bit integer words. In the case of C on the Motorola MVME147, the variable to store the buffer must be defined as a short integer.

It is important to note that command lists use screen coordinates and not virtual device coordinates. However, there appears to be a relationship between command lists and virtual devices in that the appropriate virtual device needs to be selected when a command list is created. The method used to avoid this is to select the full screen virtual device when creating and executing a command list.

### 2.2.3 Drawing Method

The graphics system provides 4 bit planes allowing a range of 16 colours per pixel. The 3 least significant bit planes are used to draw the background of the instrument, and

the fourth bit plane for the foreground. Using appropriate boolean class drawing modes and a colour palette of 16 colours, the foreground picture is modified without affecting the background information. This is achieved by setting to white all the colours in the colour table in which the foreground bit is set. This method is used to rotate the dial pointer around the dial and also to write and erase text.

The boolean class drawing mode performs the pixel combining operation on each bit position in the pixel. A result bit is computed from the corresponding source (s), and destination (d) pixel bits. The drawing mode used to draw to the foreground plane is mode 7, s OR d, and the drawing mode used to erase the foreground plane is mode 4, (NOT s) AND d. The following example illustrates this procedure.

Pixel destination (d) is green, colour 5 in colour table.  
Pixel source (s), binary value 1000.

Draw: new d = s OR d

s	1000	
d	<u>0101</u>	
new d	1101	(13, white in colour table)

Erase: new d = (NOT s) AND d

s	1000	
d	1101	
NOT s	0111	
d	<u>1101</u>	
new d	0101	(5, green in colour table)

## 2.2.4 Program Structure

The flight instrument software is written in C, in order to use the C graphics library provided by NOVA\*CGI. Each of the instruments is treated as a separate entity and is contained in a separate module. There are a number of functions common to all the instruments, such as moving the dial pointer (Move\_Pntr()), and these are located in their own module. Through the use of this methodology, it was possible to develop each instrument to completion without any effect from another instrument.

Each of the individual instrument modules has at least the three primary functions Init\_XXXX(), Update\_XXXX(), Draw\_XXXX\_Back(), where XXXX specifies the appropriate instrument. The function Init\_XXXX() is the first function called for the instrument, and it draws the instrument background by calling function Draw\_XXXX\_Back(). Init\_XXXX() also initialises the dial pointers and dial readings for the instruments. The third primary function, Update\_XXXX, is called every time the instrument data is updated, and is passed a pointer to the structure containing the instrument data.

In designing each module, the graphics functions and the mathematical model of the instrument have been separated as much as possible, in order to allow the code to be ported to another machine.

Appendix A contains a list of the modules and the functions contained in each module.

## 2.2.5 Example File

A file, instr\_example.c, contains the code required to set up and initialise the graphics system and the instruments. The highlighted section of the code is that which must be modified in order to pass the correct parameters to the Update functions.

### 2.2.6 Error Logging

A file, errorlog, is created on the initialisation of the program. If a value passed to an Update function is outside the allowable range of the instrument, a note is made in file errorlog. The error output is performed in function error(), in module stddraw.c. The time, date and the erroneous data value are written to the file.

## 3.0 INSTRUMENTS

Each instrument was designed as a separate module. This section explains each individual module, detailing any differences between modules. Common to all the instruments is the method by which the dynamic portions of the instrument are produced.

### 3.1 Airspeed Indicator - airspeed.c

The airspeed indicator is situated on the left lower portion of the right lower panel of the main instrument panel. This instrument provides backup uncompensated airspeed information in knots indicated airspeed.

The dial face information, including tick mark locations, are taken from Reference [6]. The location of the tick marks (i.e. scale) in Reference [6] differs from those in a number of photographs of the current F/A-18 cockpit. In the event that the layout of the dial face requires modification, the code can be modified by changing the array variable as\_theta[], to utilise the new angles. In addition, the constants contained in array deg\_kt[] used in calculating the dial pointer angle will require modification.

The conversion of the airspeed to a dial pointer angle is performed in the function Arspd\_Angle(). A piecewise linear function is used to model the instrument characteristics.

The airspeed information is passed to function Update\_Airspeed() as a pointer to a structure variable of type AIRSPEED, defined as:

```
typedef struct airspeed_indicator
{
    int status;           /* -1 frozen, 0 off, 1 on */
    float IAS;           /* 0 to 850 knots */
} AIRSPEED;
```

The status member is defined in all the instruments and allows the instrument to be in one of three states:

- i) a 'frozen' state where the instrument indicates the airspeed before it froze;
- ii) an 'off' state where the instrument indicates a zero value;
- iii) an 'on' state where the instrument functions normally indicating the current airspeed.

### 3.2 Altimeter - altimeter.c

The altimeter is situated on the lower centre portion of the right lower panel of the main instrument panel. The instrument provides backup pressure altitude in feet.

The dial face information is taken from Reference [6]. There exists a linear relationship between the dial pointer angle and the altitude, and this makes the conversion a

straightforward process. The function Alt\_Angle() converts the altitude (hundreds portion) to an angle in degrees.

The altitude information is passed to function Update\_Altimeter() as a pointer to a structure variable of type ALTITUDE, defined as:

```
typedef struct altimeter
{
    int status;
    float altitude;           /* 0 to 99999 feet */
    float baro_press;         /* 0 to 9999 mba */
} ALTITUDE;
```

### 3.3 Rate of Climb Indicator - climbrate.c

The rate of climb indicator is situated on the right lower portion of the right lower panel of the main instrument panel. This instrument provides backup vertical velocity in thousands of feet/min, based on barometric data.

The dial face information is taken from Reference [6]. The scale over the entire range is non-linear, however a piecewise linear function approximates the scale. The function Roc\_Angle() converts the climb rate to an angle in degrees.

A 4th order polynomial was fitted to the curve of angle (measured from 0 ft/min) versus climb rate, but this did not result in any improvement in the operating speed of the instrument. The function Roc\_Angl\_4th\_Order() contains the code to convert the climb rate to an angle using the 4th order polynomial.

The climb rate information is passed to function Update\_RateofClimb() as a pointer to a structure variable of type CLIMBRATE, defined as:

```
typedef struct variometer
{
    int status;
    float ROC;                /* -6000 to 6000 ft/min */
} CLIMBRATE;
```

### 3.4 Attitude Indicator - horizon.c

The attitude indicator is situated on the upper left of the right lower panel of the main instrument panel. The instrument provides the backup aircraft attitude, rate and direction of turn, and indicates lateral and vertical position error (ILS).

A number of difficulties were encountered in attempting to model the attitude indicator resulting in a non-functional instrument. The slow update rate of the graphics package, in particular the polygon filling commands. The inability to set up a circular window to clip objects outside the circular region. This is achieved in the IRIS version with z-buffering. The lack of C functions to perform transformations with regular graphics output<sup>3</sup>.

C functions were written to perform the transformation (rotation and translation) of the 2D data, and also to clip data against a circular window. However, the speed of the Circular Arc Centre Close function (Ref. [1] p. 85) significantly decreases the performance of the instruments. Including the function Update\_Horizon() decreases the

---

<sup>3</sup> NOVA\*CGI segments contain transformation entries in the segment state list (Ref. [1]).

update rate from 7.7 Hz to 3.5 Hz (i.e. increases the time to complete one iteration from 130 ms to 287 ms). (Refer to Table 1, Section 4.1)

The inclusion of the attitude indicator instrument reduces performance to an unacceptable level and therefore the instrument is at present non-functional. It may be possible to improve the attitude indicator by using an incremental updating method, i.e. only redrawing the portion of the screen that requires modification.

### 3.5 Jettison Station Select - jetstat.c

This instrument is situated on the left lower portion of the left lower panel of the main instrument panel. The instrument provides selection of stations for selective jettison, however this is not functional on the simulator displays. The instrument also provides the status of landing gear and flap positions.

The three indicator lights for the landing gear indicate when the nose, left and right landing gear are down and locked. The three indicator lights for the flaps indicate when the flaps are in the half down position, full down position and any other position apart from half and full.

The information is passed to function Update\_Jettison() as a pointer to two structure variables of types UNDRCRG, and FLAPS, defined as:

```
typedef struct undercarriage
{
    int status;
    int nose_down_locked;           /* 0 unlocked, 1 locked */
    int left_down_locked;           /* 0 unlocked, 1 locked */
    int right_down_locked;          /* 0 unlocked, 1 locked */
} UNDRCRG;

typedef struct flps
{
    int status;
    int flaps_half;                 /* 0 off, 1 on */
    int flaps_full;                 /* 0 off, 1 on */
    int flaps_alert;                /* 0 off, 1 on */
} FLAPS;
```

### 3.6 Engine - engine.c

The engine readings instrument is situated on the centre of the lower left panel of the main instrument panel. The instrument provides left and right engine information on rotor speed, exhaust gas temperature, fuel flow, nozzle position and oil pressure.

The relationship between nozzle position and dial pointer angle is a linear function and the conversion of the nozzle position to an angle is performed in function Noz\_Angle().

The engine data is passed to function Update\_Engine() as a pointer to a structure variable of type ENGINES, defined as:

```
typedef struct engine_status
{
    int status;
    float RPM;                      /* engine rotor speed 0 to 110% */
    float EGT;                      /* exhaust gas temperature 0 to 999°C */
}
```

```

float FF;                /* fuel flow 0 to 15000 lbs/h */
float NOZ;               /* nozzle position 0 to 100 % */
float OIL;               /* oil pressure 0 to 200 psi */
} ENG_STATUS;

typedef struct engs
{
    ENG_STATUS left;
    ENG_STATUS right;
} ENGINES;

```

### 3.7 Fuel - fuel.c

The fuel quantity instrument is situated on the right of the lower left panel of the main instrument panel. This instrument provides readings of total internal fuel, total fuel (internal and external), and a left and right reading depending on the position of the rotary selector.

The rotary selector is non-functional and the instrument will display the value passed in structure members left and right (refer to structure definition). The BINGO index bug value is passed in the structure variable and cannot be selected by the pilot.

The relationship between total internal fuel and the dial pointer angle is linear and the conversion of the value to an angle is performed in function Fuel\_Angle().

The fuel data is passed to Update\_Fuel() as a pointer to a structure variable of type FUEL, defined as:

```

typedef struct fuel_info
{
    int status;
    float internal_total;    /* total internal fuel 0 to 12000 lbs */
    float intext_total;     /* total internal & external fuel
                           0 to 99999 lbs */
    float left;             /* left readout 0 to 9999 */
    float right;            /* right readout 0 to 9999 */
    float BINGO;            /* BINGO index bug position 0 to 12000 */
    int off;                /* 0 off, 1 on */
    int id;                 /* 0 off, 1 on */
} FUEL;

```

The structure members off and id are related to the two indicators on the instrument. The off indicator indicates that power is not being applied to the fuel gauge system, and the id indicator indicates a failed intermediate device (see Ref. [6] for detailed description).

### 3.8 Porting Code to IRIS

Software emulations of the instruments located on the right lower panel of the main instrument panel have been ported to the IRIS. The files have \_iris appended to their generic names, e.g. IRIS version of airspeed.c is airspeed\_iris.c. The graphics routine function calls used by NOVA\*CGI are replaced by IRIS GL functions. The process is not difficult and the remainder of the code could very easily be ported to the IRIS.

The performance increased by a factor of 10, as would be expected. The dial pointers are drawn in the frontbuffer, instead of the bitplaning method used in the CGI version.

## 4.0 PERFORMANCE

### 4.1 Dial Pointer Drawing

In the early development of the software it became apparent that the dial pointer drawing algorithm would be the major factor influencing the performance of the instruments. Initially a solid pointer was drawn using the NOVA\*CGI functions but this resulted in a very slow update rate. The same pointer drawn with a hollow fill style attribute was implemented, leading to an improvement in the update rate. The third option investigated was a pointer made up of a line and a solid triangular tip, but this resulted in a reduction in update rate by a factor of 2. (Table 1)

The ideal pointer geometry is a solid filled pointer, but a large time penalty results from the use of the high level graphic functions. The NOVA\*CGI device level command list performance extensions, included in the Version 3.2 upgrade, provided the ability for drawing the pointers as solid filled objects, without incurring a performance reduction.

A command list for every possible pointer position is stored in the CGI Interpreter's local memory, to be executed at a later stage via the command list identifier. The limit of 1024 command list identifiers resulted in only being able to draw a pointer at 30 increments. Due to the reduced command set available to the user, a command to draw a filled polygon does not exist. The only option available is to use the CL Fill Trapezoid command.





Instrument	Dial Pointer					CL 		CL 	
		ms	Hz	ms	Hz	ms	Hz	ms	Hz
Airspeed, Altimeter, Climbrate	Version 2.0 <sup>4</sup>	34	29.4	63	15.9				
	Version 3.2	34	29.4	63	15.9				
All Instruments	Version 2.0	172	5.8	217	4.6				
	Version 3.2	152	6.6			120	8.3	130 <sup>5</sup>	7.7

Table 1: Flight Instrument Performance

### 4.2 Trapezoidal Decomposition of a Polygon

The command list provides a Fill Trapezoid command that allows a solid trapezoid to be drawn. However, the base and top edge of the trapezoid must be parallel to the conventional x-axis. An algorithm (Ref. [7]) to decompose an arbitrary polygon into a number of trapezoids is used to create the pointer drawing command list buffer. Appendix B contains an illustrated example of the algorithm. The algorithm is:

Step 1: Select one of the equal highest vertices of the polygon and determine adjacent vertices with the same height. Let vertices  $V_a$  and  $V_b$  from the selected vertices define the top edge of a component trapezoid. Vertices  $V_a$  and  $V_b$  are coincident if a single highest vertex exists.

Step 2: Let vertex  $V_c \neq V_a$  be adjacent to  $V_b$  and  $V_d \neq V_b$  be adjacent to  $V_a$ .

<sup>4</sup> Version number refers to the version of NOVA\*CGI.

<sup>5</sup> The difference of 10 ms between this time and the time for the hollow command list pointer is due to the inclusion of the updating of the jetison station select instrument.

Step 3: (Used in concave polygon case)

Step 4: Remove component trapezoid  $V_a, V_b, V_f, V_g$  from the polygon where  $V_f$  = intersection of line  $(V_b, V_c)$  with  $y = y_m$ ,  $V_g$  = intersection of line  $(V_a, V_d)$  with  $y = y_m$  and  $y_m = \max(y_c, y_d)$ .

Step 5: Return to Step 1 and repeat until all component trapezoids are removed.

The algorithm is implemented using a circular doubly linked list of vertices to model the polygon. Each vertex is defined as:

```
struct vertices
{
    int x;                /* x coordinate */
    int y;                /* y coordinate */
    struct vertices *next; /* pointer to next vertex */
    struct vertices *prior; /* pointer to prior vertex */
}
```

The module cltrap.c contains the functions that are used to perform the trapezoidal decomposition. This method has resulted in a solid pointer being drawn, and a considerable increase in the performance.

#### 4.3 Overall Performance

It is important to note that as each instrument is added to build up the flight instrument panels the performance is reduced. This is particularly evident with the engine instrument, because it contains a high number of text output functions in every update of the instrument. One possible method to improve this is to convert the text writing portions of the program to use the command list text output operation CL Text.

The performance values given in Table 1 are for three instruments, and all the instruments. This is the worst case scenario, in which all values are constantly changing, leading to continual graphics updates.

#### 5.0 CONCLUSION

The left and right lower panel backup flight instruments described in this document were modelled and displayed on two raster displays for the F/A-18 flight simulator. The software is written in C using NOVA\*CGI, and is running on a Motorola MVME147 single board computer, under the UNIX System V/68 operating system. A version of the software was ported to the IRIS, demonstrating the ease with which this can be performed.

The modelling of the attitude indicator has not been achieved on the Motorola version, due to inadequate performance of the graphics system. This may be improved by later versions of NOVA\*CGI, leading to a functional attitude indicator.

#### ACKNOWLEDGEMENTS

The author would like to thank Mr Cameron Lewis for his support throughout the development of the software, and Mr Gerald Sterling and Mr Graeme Saleeba for their contributions in preparation of this document.



## REFERENCES

- [1] Nova Graphics International. *NOVA\*CGI Reference Manual Version 3.2*. Austin, Texas, USA. 15 April 1990.
- [2] ISO/IEC DIS 9639. *Information Technology - Computer Graphics - Interfacing for Dialogues with Graphical Devices*. USA. 6 March 1990.
- [3] Nova Graphics International. *NOVA\*CGI and the MVME393 Boards*. Austin, Texas, USA. November 1990.
- [4] Motorola Inc. *MVME147 MPU VME module and MVME712/MVME712M Transition Module. User's Manual*. Phoenix, Arizona, USA. April 1988.
- [5] Motorola Inc. *MVME393 Multi-Channel Graphics Display Controller User's Manual*. Phoenix, Arizona, USA. June 1988.
- [6] McDonnell Aircraft Company. *F/A-18 and TF/A-18 Human Engineering Crew Station Design Document Report MDC A4277-4*. Saint Louis, Missouri, USA. 22 February 1979.
- [7] Little W.D. and Heuft R. *An Area Shading Graphics Display System*. IEEE Transactions on Computers, c-28, 7, 528-531, 1979.

## **APPENDIX A - List of Modules and Functions<sup>6</sup>**

**Module: defs.h**

**Module: airspeed.c**

**Functions:**

**Init\_Airspeed()**

**Update\_Airspeed()**

**Draw\_ArSpd\_Back()**

**Arspd\_Angle()**

**Module: altimeter.c**

**Functions:**

**Init\_Altimeter()**

**Update\_Altimeter()**

**Draw\_Alt\_Back()**

**Alt\_Angle()**

**Change\_Alt()**

**Update\_Baro\_Press()**

**Module: climbrate.c**

**Functions:**

**Init\_RateofClimb()**

**Update\_RateofClimb()**

**Draw\_ROC\_Back()**

**Roc\_Angle()**

**Roc\_Angle\_4th\_Order()**

**Module: jetstat.c**

**Functions:**

**Init\_Jettison()**

**Draw\_Jet\_Back()**

**Update\_Jettison()**

**Module: engine.c**

**Functions:**

**Init\_Engine()**

**Draw\_Eng\_Back()**

**Update\_Engine()**

**update\_cntr()**

**Noz\_Angle()**

---

<sup>6</sup>The function names do not include the parameter arguments passed to each function.

**Module: fuelqty.c**

**Functions:**

Init\_Fuel()  
Draw\_Fuel\_Back()  
Draw\_Bingo()  
Bingo\_Coords()  
Update\_Fuel()  
update\_fl1\_cntr()  
update\_fl2\_cntr()  
update\_indicator()  
Fuel\_Angle()

**Module: horizon.c**

**Functions:**

Init\_Horizon()  
Draw\_Horiz\_Back()  
Update\_Horizon()  
Pitch\_Dispatch()  
make\_identity()  
combine\_transformations()  
rotate()  
translate()  
transform\_points()  
Circle\_Clip\_Lines()  
In\_Out\_Circle()  
Intersect\_Line\_Circle()

**Module: stddraw.c**

**Functions:**

Instr\_Brdr()  
Draw\_Ticks()  
error()

**Module: cldraw.c**

**Functions:**

Move\_Pntr()  
Roundoff3()  
Create\_Pntr\_CmdLists()  
Pntr\_Coords()

**Module: cltrap.c**

**Functions:**

Trapezoid\_Decomposition()  
store\_vertex()  
intersect\_lp()  
fill\_trapezoid\_buffer()

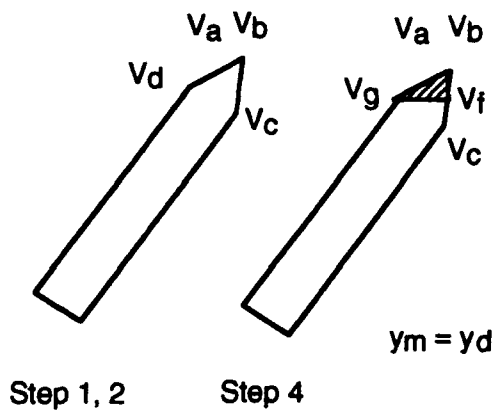
**Module: istup4.c**

**Functions:**

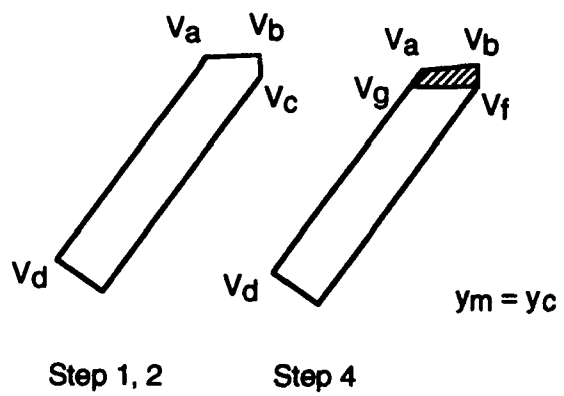
InitialiseCGI()  
TerminateCGI()

## APPENDIX B - Trapezoidal Decomposition of a Dial Pointer

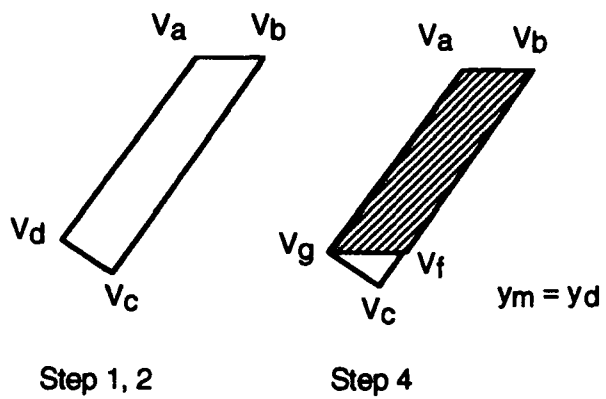
### Iteration 1



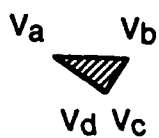
### Iteration 2



**Iteration 3**



**Iteration 4**



## FIGURES

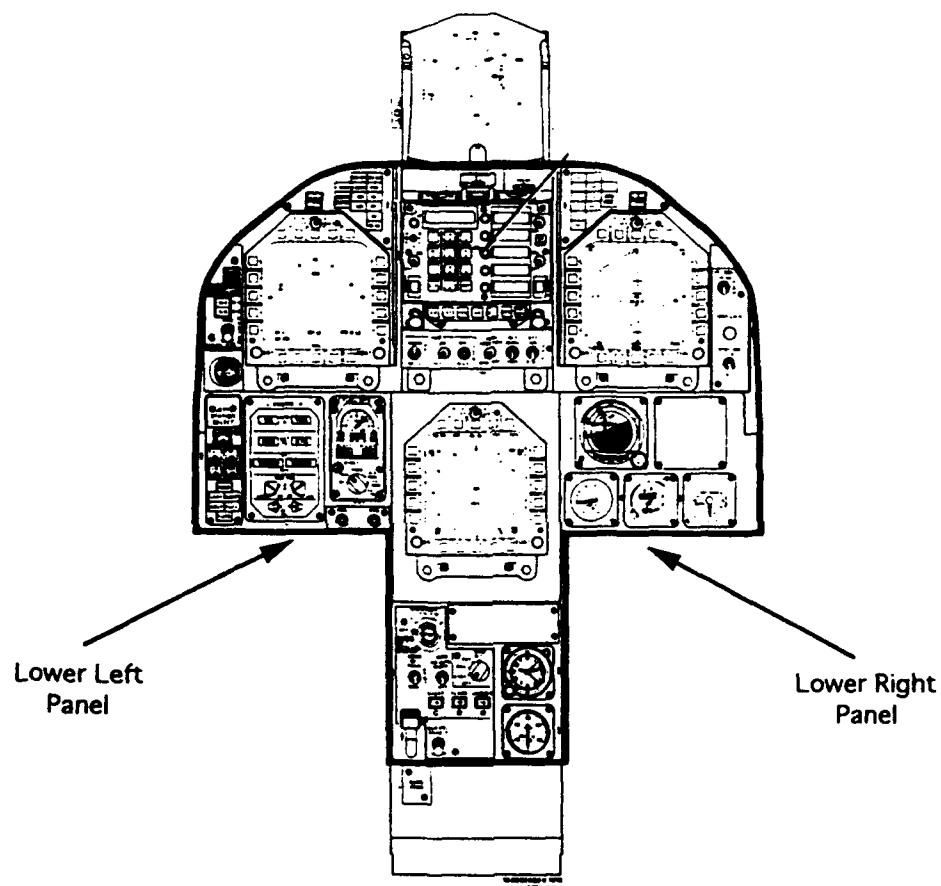


Figure 3 - F/A-18 Main Instrument Panel

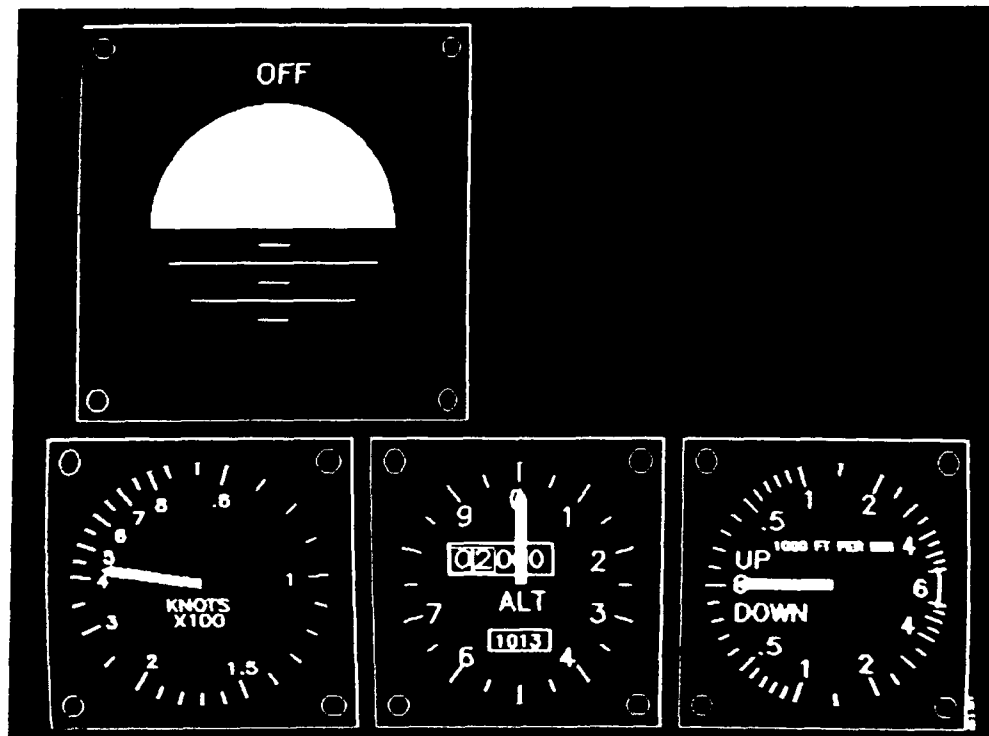
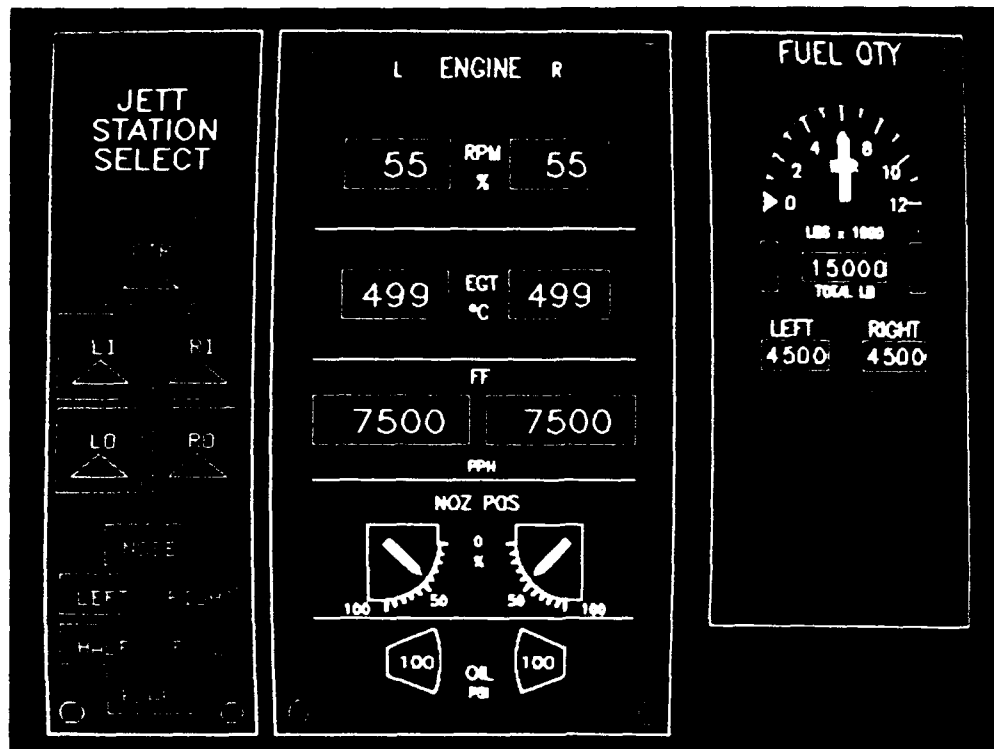


Figure 4 - Flight Instrument Displays

## DISTRIBUTION

### AUSTRALIA

#### Department of Defence

##### Defence Central

Chief Defence Scientist	}	shared copy
AS Science Corporate Management		
FAS Science Policy		
Director Departmental Publications		
Counsellor Defence Science, London (Doc Data sheet only)		
Counsellor Defence Science, Washington (Doc Data sheet only)		
Scientific Adviser, Defence Central		
OIC TRS, Defence Central Library		
Document Exchange Centre, DSTIC (8 copies)		
Defence Intelligence Organisation		
Librarian Defence Signals Directorate, (Doc Data sheet only)		

##### Aeronautical Research Laboratory

Director  
Library  
Chief Air Operations Division  
Author: Y.Y. Link (2 copies)  
K. Anderson  
C. Lewis  
G. Sterling  
G. Saleeba  
N. Matheson  
S. Jobson

##### Defence Science & Technology Organisation - Salisbury Library

##### Navy Office

Navy Scientific Adviser (3 copies Doc Data sheet only)

##### Army Office

Scientific Adviser - Army (Doc Data sheet only)

##### Air Force Office

Air Force Scientific Adviser (Doc Data sheet only)  
OIC ATF, ATS, RAAFSTT, WAGGA (2 copies)



Universities and Colleges

Sydney  
Engineering Library  
Head Aeronautical Department

NSW  
Head Aerospace Engineering

RMIT  
Head Aerospace Engineering

SPARES ( 4 COPIES)

TOTAL (34 COPIES)

## DOCUMENT CONTROL DATA

PAGE CLASSIFICATION  
UNCLASSIFIED

PRIVACY MARKING

1a. AR NUMBER AR-006-680	1b. ESTABLISHMENT NUMBER ARL-TN-17	2. DOCUMENT DATE MARCH 1993	3. TASK NUMBER DST 91/104
4. TITLE  FLIGHT INSTRUMENT SOFTWARE FOR THE F/A-18 RESEARCH SIMULATOR		5. SECURITY CLASSIFICATION (PLACE APPROPRIATE CLASSIFICATION IN BOX(S) IE: SECRET (S), CONF. (C) RESTRICTED (R), UNCLASSIFIED (U)).	6. NO. PAGES  20
		<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 2px; text-align: center;">U</div> <div style="border: 1px solid black; padding: 2px; text-align: center;">U</div> <div style="border: 1px solid black; padding: 2px; text-align: center;">U</div> </div> <div style="display: flex; justify-content: space-around; font-size: small;"> <span>DOCUMENT</span> <span>TITLE</span> <span>ABSTRACT</span> </div>	7. NO. REFS.  7
8. AUTHOR(S)  Y.Y. LINK		9. DOWNGRADING/DELETING INSTRUCTIONS  Not applicable.	
10. CORPORATE AUTHOR AND ADDRESS  AERONAUTICAL RESEARCH LABORATORY  AIR OPERATIONS DIVISION  506 LORIMER STREET  FISHERMENS BEND VIC 3207		11. OFFICE/POSITION RESPONSIBLE FOR:  <div style="text-align: center;">DSTO</div> SPONSOR _____ SECURITY _____ DOWNGRADING _____ <div style="text-align: center;">CAOD</div> APPROVAL _____	
12. SECONDARY DISTRIBUTION (OF THIS DOCUMENT)  Approved for public release.  OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DSTIC, ADMINISTRATIVE SERVICES BRANCH, DEPARTMENT OF DEFENCE, ANZAC PARK WEST OFFICES, ACT 2601			
13a. THIS DOCUMENT MAY BE ANNOUNCED IN CATALOGUES AND AWARENESS SERVICES AVAILABLE TO ....  No limitations.			
13b. CITATION FOR OTHER PURPOSES (IE. CASUAL ANNOUNCEMENT) MAY BE			
<input checked="checked" type="checkbox"/> UNRESTRICTED OR		<input type="checkbox"/> AS FOR 13a.	
14. DESCRIPTORS F/A-18 Aircraft flight simulator and visual system Flight instruments Computer graphics Software engineering		15. DISCAT SUBJECT CATEGORIES 0108 0104 010303	
16. ABSTRACT <i>The standby flight instruments are replicated on two high resolution raster displays for the F/A-18 research simulator. A software package was developed using NOVA*CGI, an implementation of the CGI graphics standard, to control a MVME393 multi-channel graphics controller. The software is written in C for the Motorola MVME147 single board computer under the UNIX System V/68 operating system. Increased performance is achieved by decomposing the instrument dial pointers into a number of trapezoids, and storing these as device level command lists.</i>			

PAGE CLASSIFICATION  
**UNCLASSIFIED**  
PRIVACY MARKING

THIS PAGE IS TO BE USED TO RECORD INFORMATION WHICH IS REQUIRED BY THE ESTABLISHMENT FOR ITS OWN USE BUT WHICH WILL NOT BE ADDED TO THE DISIS DATA UNLESS SPECIFICALLY REQUESTED.

16. ABSTRACT (CONT).

17. IMPRINT

**AERONAUTICAL RESEARCH LABORATORY, MELBOURNE**

18. DOCUMENT SERIES AND NUMBER

Technical Note 17

19. WA NUMBER

72 7101

20. TYPE OF REPORT AND PERIOD COVERED

21. COMPUTER PROGRAMS USED

22. ESTABLISHMENT FILE REF.(S)

23. ADDITIONAL INFORMATION (AS REQUIRED)